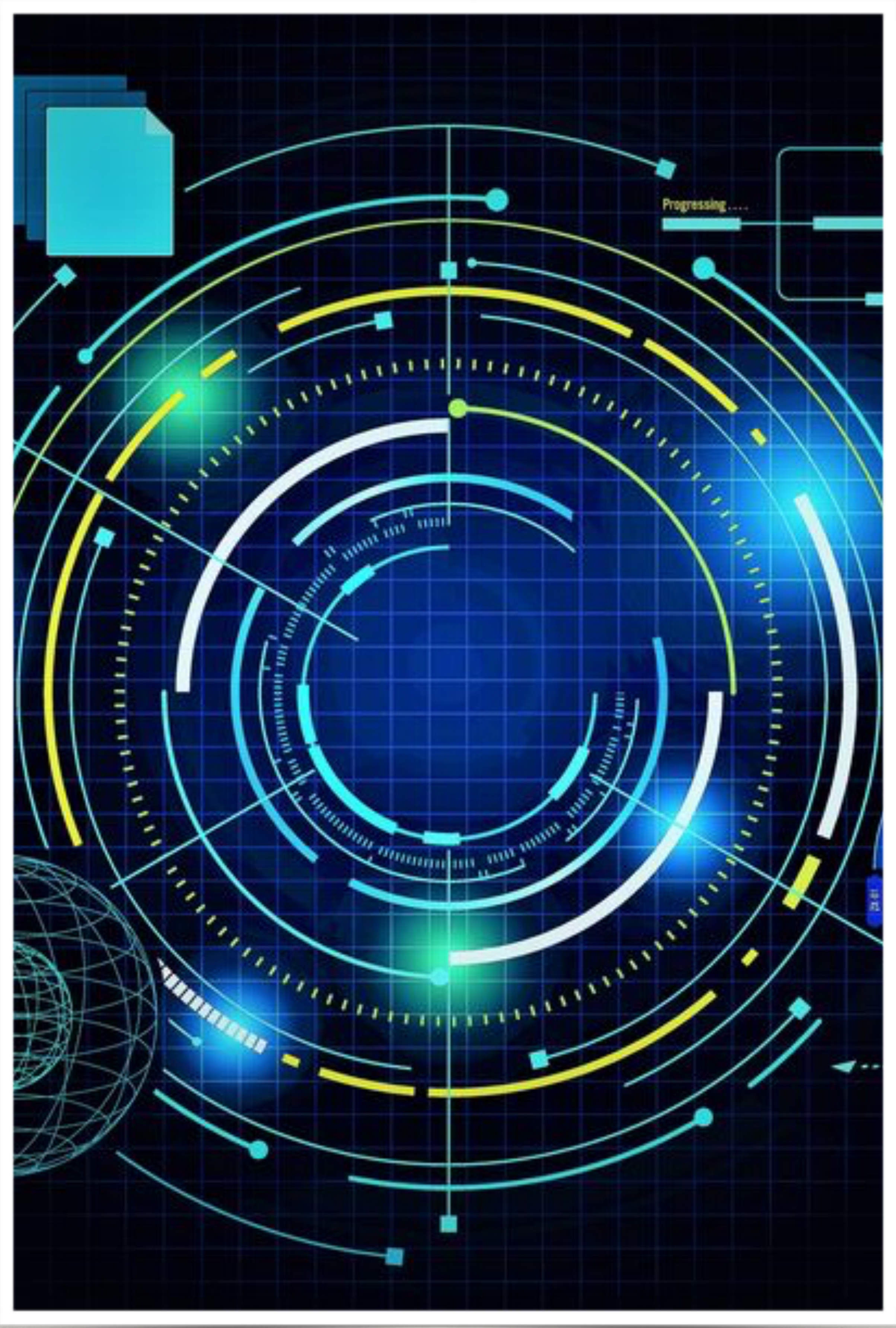
Postgraduate course
Universitat de Valencia 2020

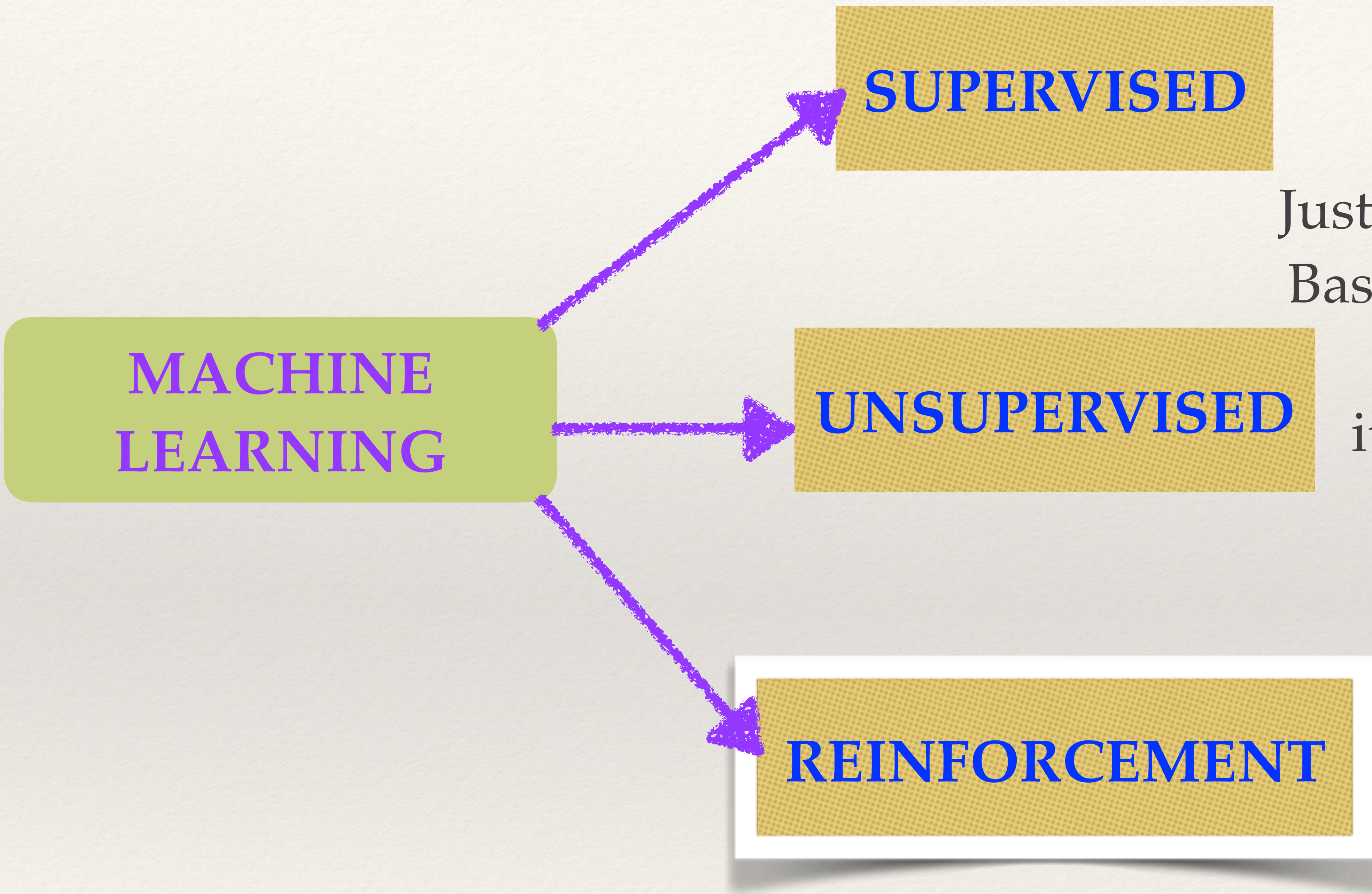
Introduction to Machine Learning for physicists

Veronica Sanz (UV/IFIC)

LECTURE 5
REINFORCEMENT



Types of learning



Just touched the surface
Basis to explore further
and incorporate
it in your research

Today's lecture

Supervised to Reinforced Learning

We have not covered important techniques like
VAEs (Variational AutoEncoders) &
GANs (Generative Adversarial Networks)*
based on Generative models in DNNs

Cool ways to accelerate learning, capture important aspects of the data, incorporate different types of data

Learn **from** humans to do what humans **already do**, but better and faster, and in more difficult situations

But, what if we wanted
a machine to become **better** than a human
at completing a high-level task?

* See these lectures

Let's find a DIFFICULT task

A truly human-difficult task
not just a task that a machine can do faster or with lower resolution

Supervised/unsupervised learning identifies *patterns* in data
But this isn't the same as learning to develop a *strategy*
and to do it better than a human

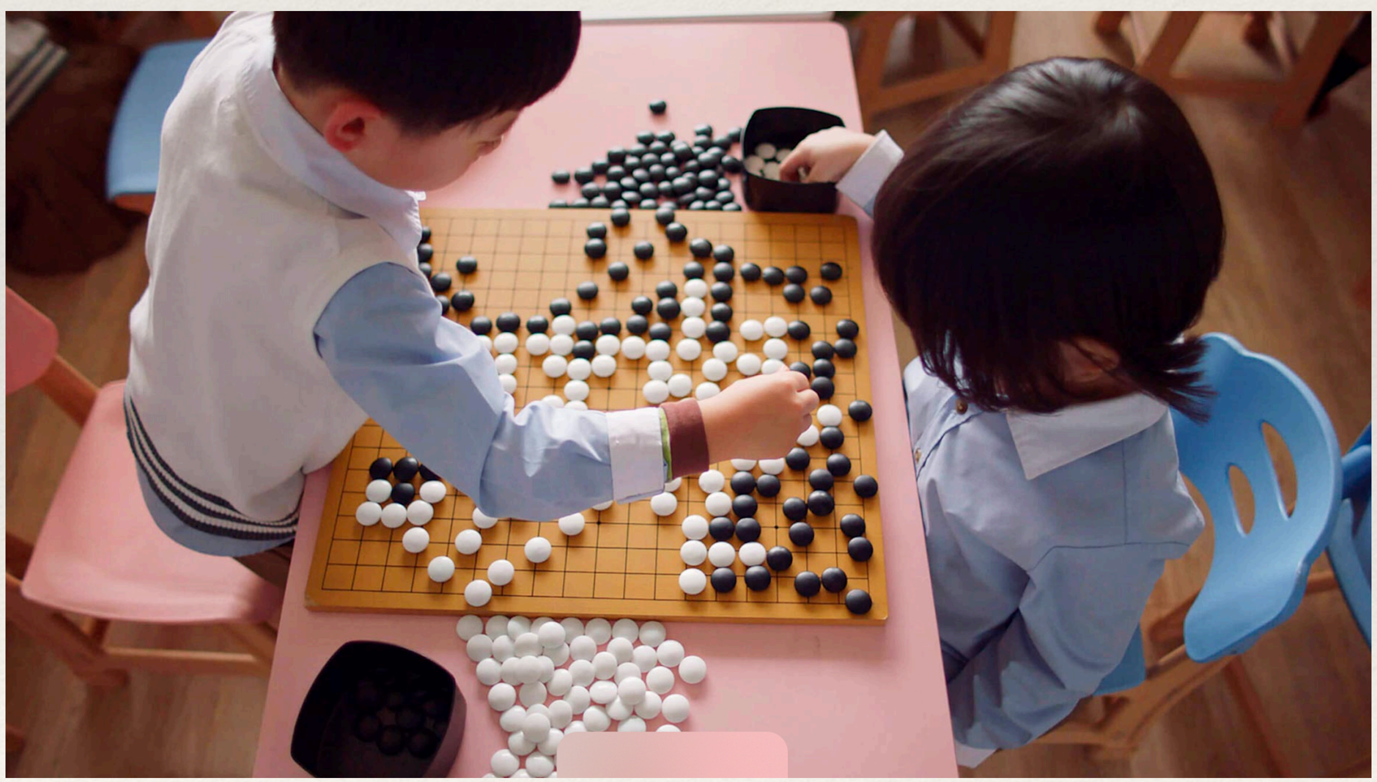
Chess is a high-level activity
different players develop different strategies
the goal is *long-term*
important pieces can be sacrificed to achieve
checkmate some moves along the way
and you have an adversary which will oblige
you to *reassess* your strategy at each step
combinatorics is ginormous

Human vs Machine

February 1996

Deep Blue (IBM) beat Garry Kasparov (World Champion) and did it again many times after brute-force computing power analysing many hundreds of millions positions / second

Human vs Machine



February 1996

Deep Blue (IBM) beat Garry Kasparov (World Champion) and did it again many times after brute-force computing power analysing many hundreds of millions positions / second

October 2015

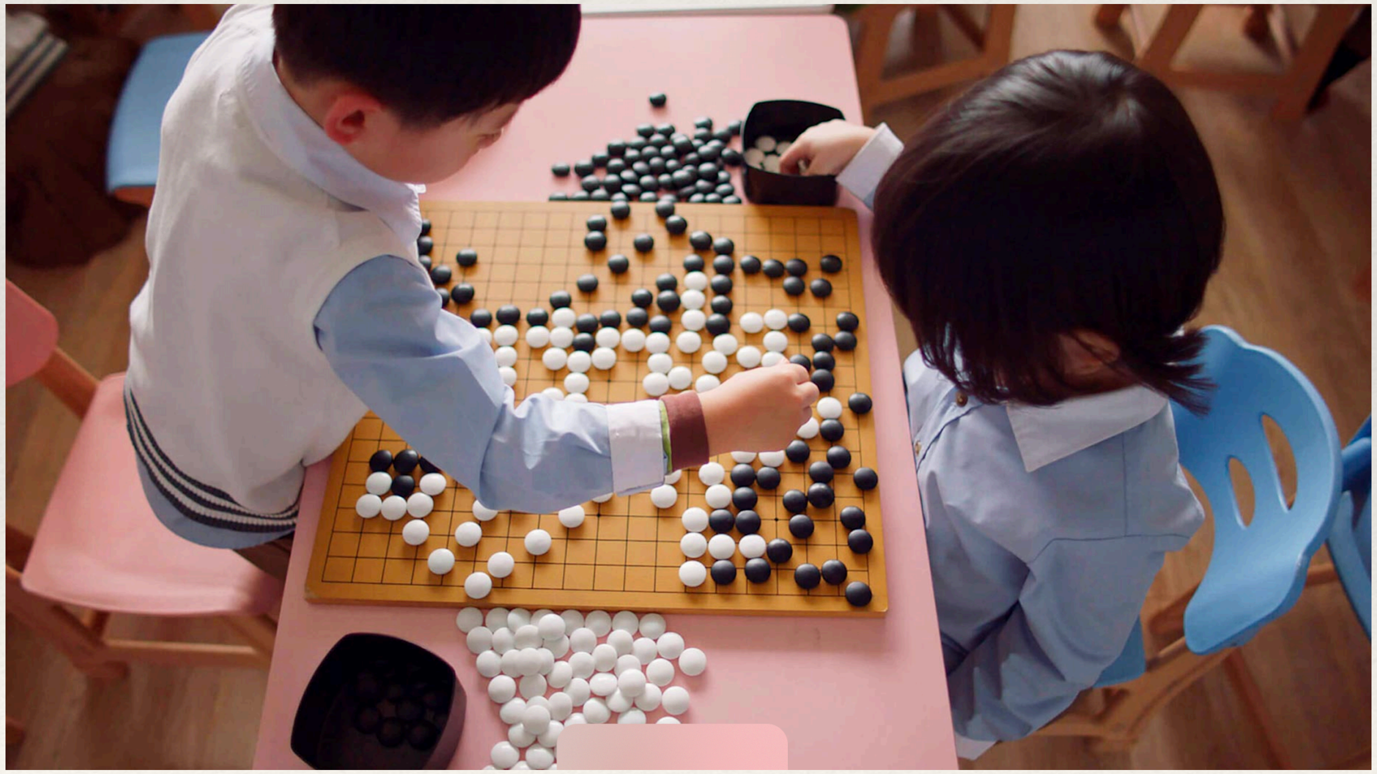
AlphaGo Zero beats a professional Go player learned from playing against *itself*

November 2017

AlphaZero builds on DNNs to beat world champions in Go, chess and shogi

Human vs Machine

February 1996
Deep Blue (IBM) beat Garry Kasparov (World Champion) and did it again many times after brute-force computing power analysing many hundreds of millions positions /second

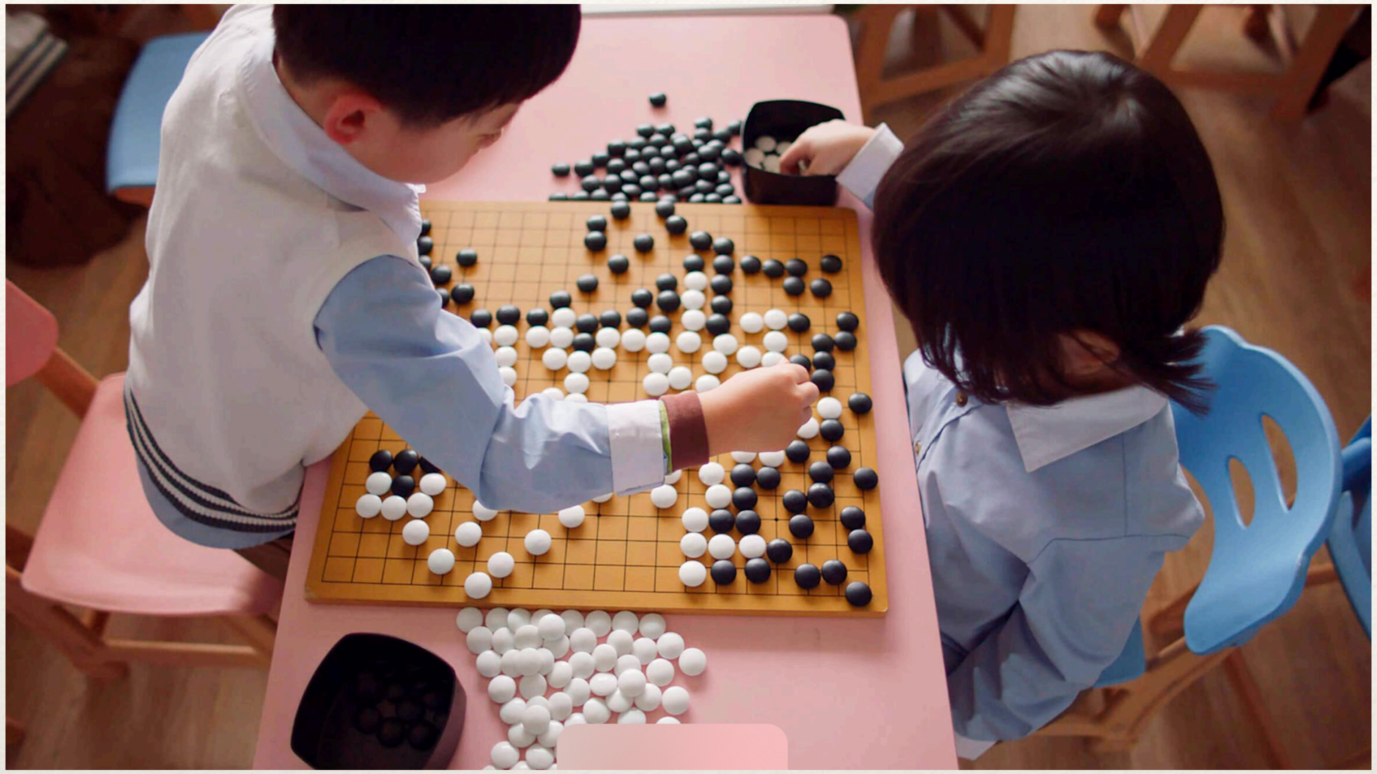


October 2015
AlphaGo Zero beats a professional Go player learned from playing against *itself*

November 2017
AlphaZero builds on DNNs to beat world champions in Go, chess and shogi

A new paradigm of learning: **REINFORCEMENT**

Go game

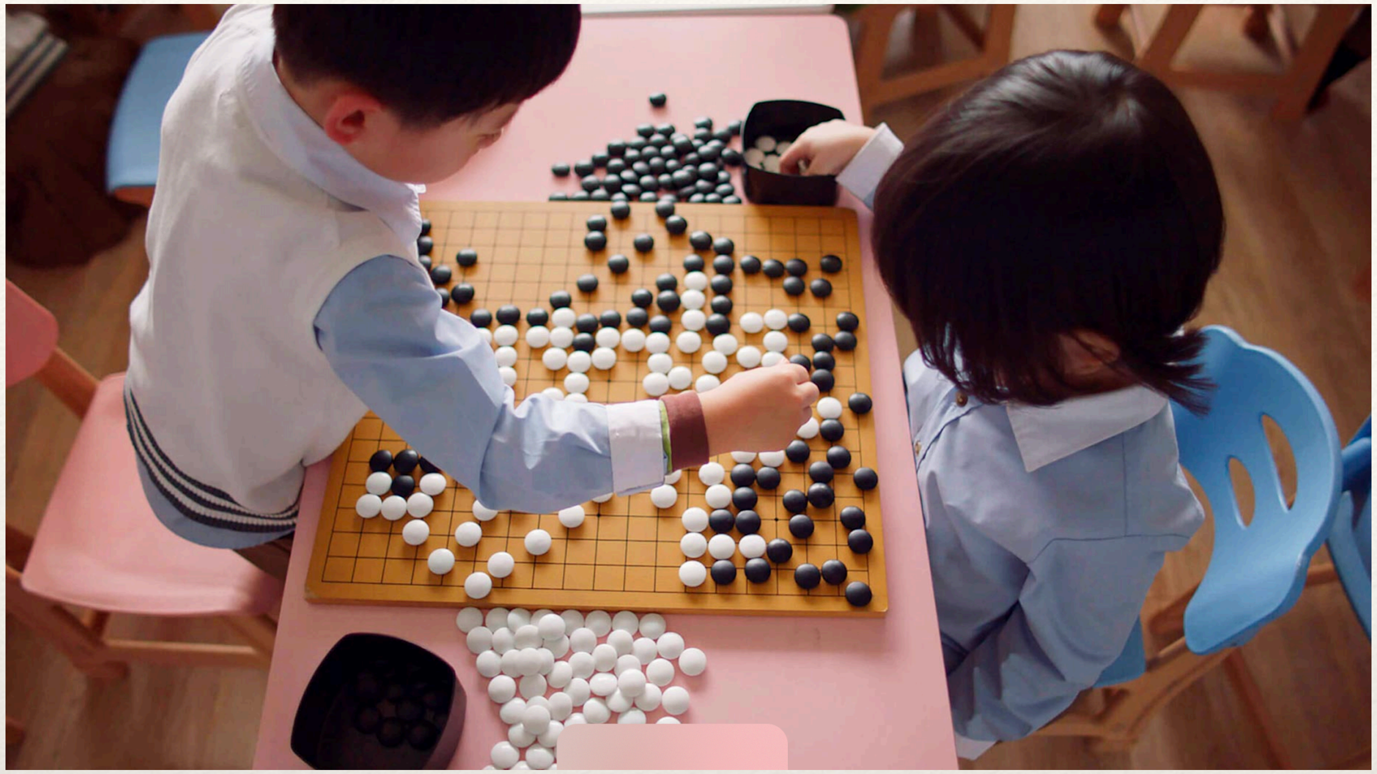


Simple game: moves are simple
no hierarchy like chess
king/queen/bishop/pawn…
goal: surround and capture
opponents' pieces

Simple rules, extreme levels of complexity when building strategies
no machine could beat a Go-master until 2015
Why is it so difficult?
how would you teach a machine to learn this game?

X,y

Go game



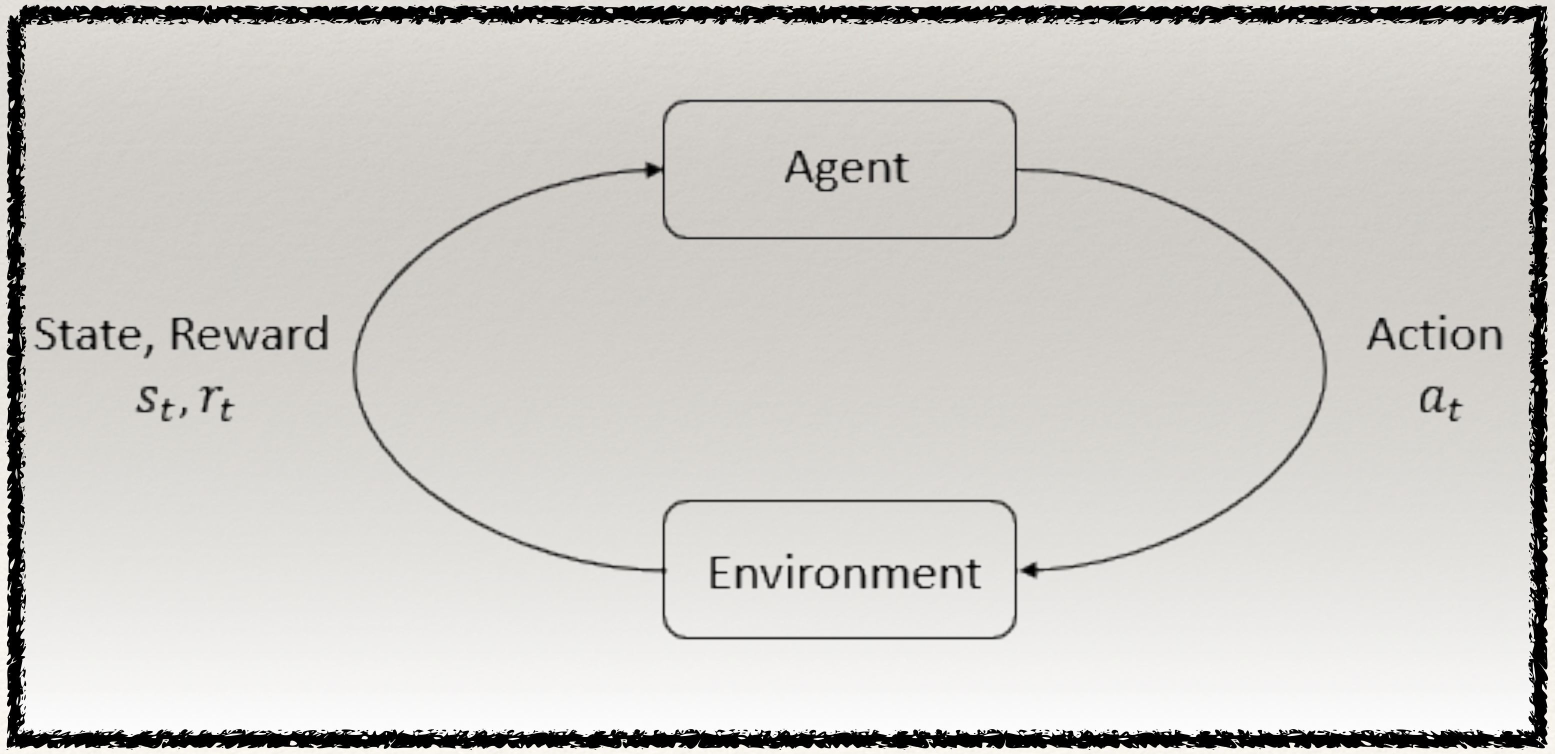
Simple game: moves are simple
no hierarchy like chess
king/queen/bishop/pawn…
goal: surround and capture
opponents' pieces

develop a strategy for long-term winning:
3^(19*19)~10^172 configurations at one step
decision in this one step guided by possible future gains
but opponent's actions change every subsequent move

Reinforcement learning

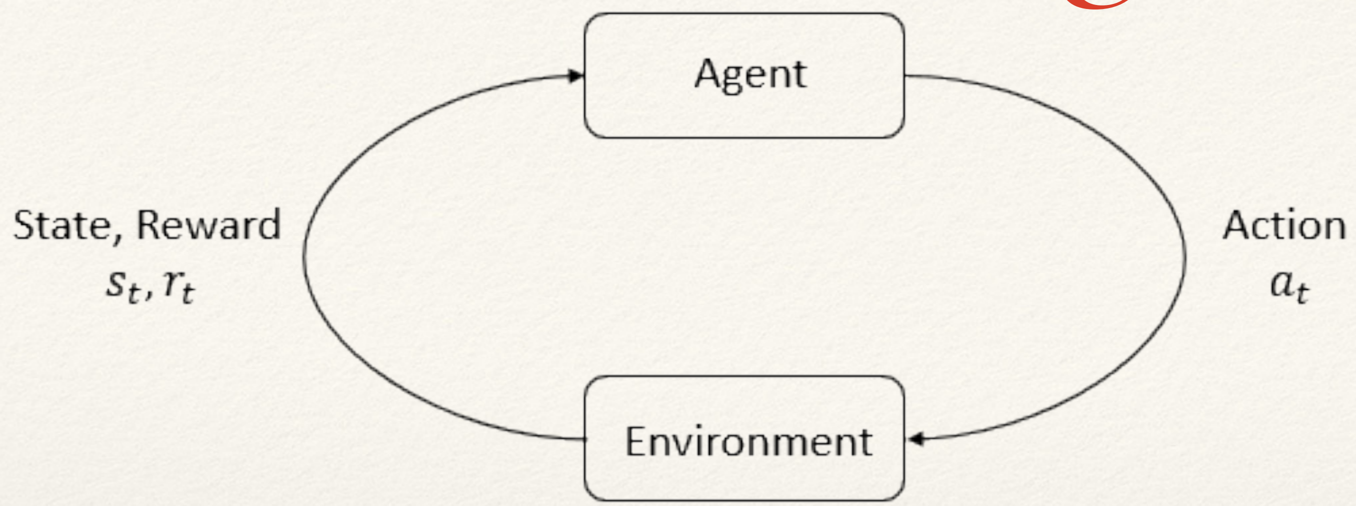
The task of getting better at Go was too difficult
too many possibilities, no human could teach from example
To beat humans we had to allow machines to learn in a different way

Machine needs to learn to make good sequences of decisions
dealing with delayed labels and developing a long-term strategy
Some form of iterative way of improving strategy
which can examine many steps ahead



agent interacts with
the **environment** in **state** *st*
takes **actions** based on **reward** *rt*
which tells about good current state is
GOAL: maximise total about of
rewards (**return**)
RL help the agent to achieve goal

Reinforcement learning: concepts



State/Observation: some kind of tensor (e.g. *an image*)

Action: possible transformation of the state (e.g. *move pawn*)

Policy: rule used by the agent to decide what action to take

$$a_t = \pi(s_t)$$

can be deterministic or schotastic and often parametrised

$$a_t = \pi_\theta(s_t)$$

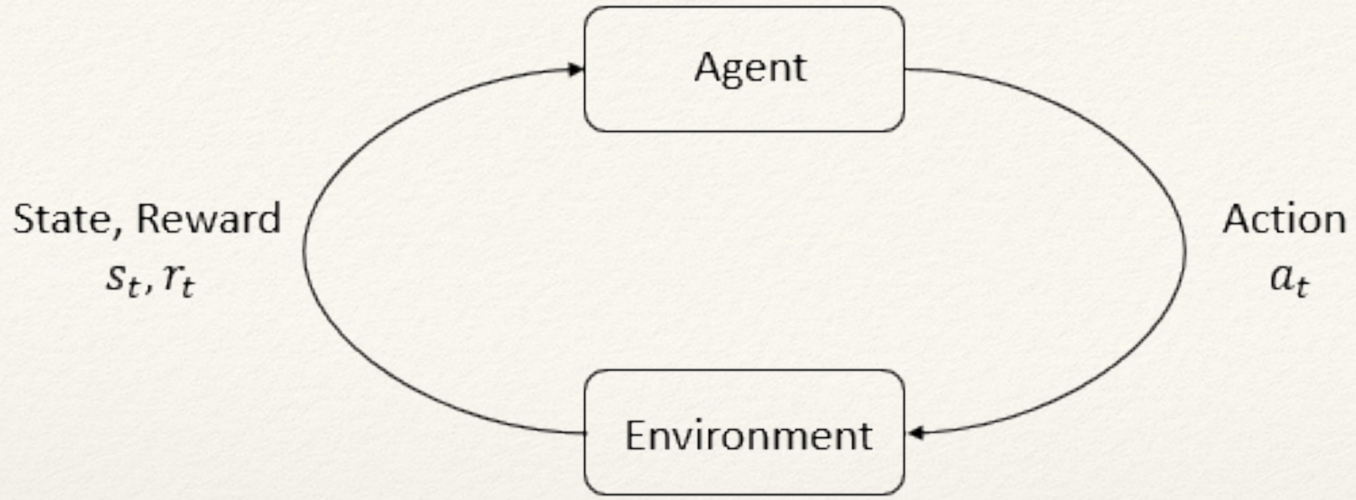
by some form of modelling of possible new situations

this sampling of possible trajectories

$$\tau(s_0,\ a_0,\ s_1,\ a_1 \ldots)$$

often done with DNNs (**Deep Reinforcement**)

Reinforcement learning: concepts



Reward: some function of current state and action taken, and next state

Return: total reward in a full sequence, a trajectory

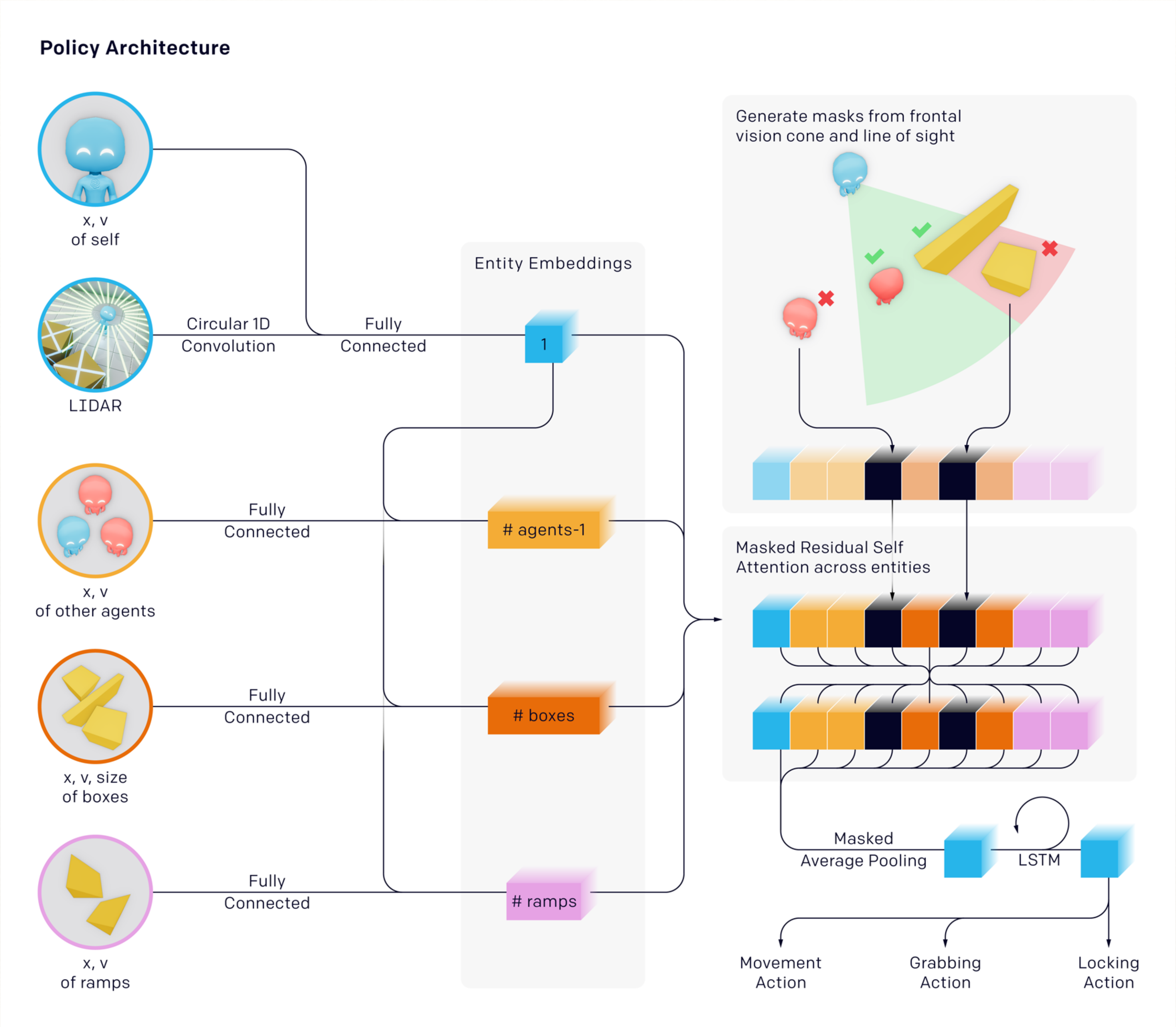
$$R(\tau) = \sum_{t=0}^{T} r_t$$

Real problems have limitations, so not all trajectories can be taken at no cost (e.g. *time limit, loss at each step…*) and we introduce a **discount**

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$$

cash now / cash in few years

Reinforcement learning: fun video



And a super fun blog and video

Reinforcement learning: overview

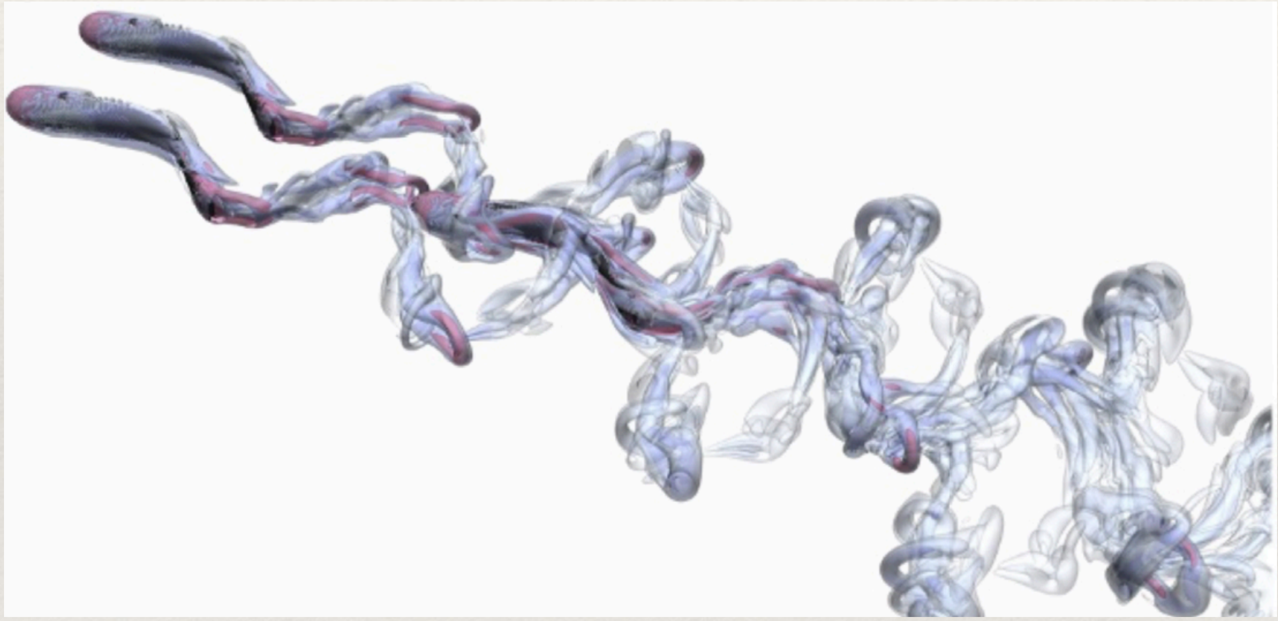
Clearly, this is a complex set-up
states could contain lots of information
an agent could choose among many actions
the number of possible steps could be very large
rewards are set to help the algorithm to increase final return
(**policy optimisation**)
but their efficiency depends on ability to explore
how the state changes by itself (opponent) or by the action
RL helps learning an *environment*

There are many, many possibilities
most successful are based on Deep Learning
& good adaptation to environment changes
e.g. ability to dynamically drop and add terms in policy

Today

We will go through a *Tensorflow* tutorial
on training agents using different policies
Environment: Cartpole (start reading this post)

In Physics, mostly unexplored
complex numerical simulations: many body, fluid dynamics…



situations with many agents
and interactions
see e.g. this nice example fish
coordinated swimming for
energy saving

Tomorrow

MORE *NON-TRIVIAL* USES:
TRANSFER LEARNING, SYMBOLIC AI,
DETECTING SYMMETRIES, NEURAL ODEs